Rapport Cryptographie - Texte A

Encadré par Pierre Vincent KOSELEFF

Réalisé par : Amine ELKARI - Mohamed-Amine CHAFIK -Zakarya JOUHAFA

Notebook Content

Introduction

Il y a à peine quelques années, le problème de la sécurité des transmissions de données semblait être l'apanage des seuls militaires. Ce n'est plus le cas avec l'essor des techniques numériques dans le commerce (Internet, les cartes de crédit, les télécommunications, etc.)

Les méthodes empiriques traditionnelles se sont révélées trop fragiles ; elles reposaient souvent sur le schéma suivant : on choisit un livre, une fois pour toutes, et on considère la permutation des vingt-six lettres de l'alphabet définie par les 26 premières lettres de ce livre.

Le codage d'un texte consiste alors à appliquer cette permutation σ au texte à coder, et le décodage à appliquer la permutation réciproque σ^{-1} au texte à décoder.

En numérique, si par exemple le message à transmettre est codé sur 64 bits, on peut employer cette technique en considérant une permutation sigma in S_{64}. C'est ce genre d'idées qui est sous-jacent aux procédés DES, dû à IBM, et qui est encore le plus utilisé en informatique.

Le talon d'Achile de ce genre de crypto système est le suivant : si une personne sait coder, il aussi décoder, car il est très facile de trouver la permutation réciproque d'une permutation sur 26, 64, 128 ou même 256 lettres. C'est pourquoi l'apparition de crypto systèmes dits à clés publique, à la fin du dernier siècle.

Cryptographie à clef secrète

Le principe de la cryptographie à clef secrète est probablement le principe le plus naturel auquel on pense lorsqu'on envisage de cacher une information : l'émetteur et le récepteur partagent un secret commun qui permet de chiffrer et de déchiffrer un texte. Les opérations pour le codage et pour le décodage sont alors essentiellement les mêmes, d'où le qualificatif « symétrique » pour de tels cryptosystèmes.

Cryptographie à clef publique

L'idée de la cryptographie à clef publique est née dans les années 1970. La personne qui va recevoir des informations publie dans un annuaire une clef qui permet de chiffrer les messages. Seule une clef privée détenue uniquement par le récepteur doit rendre possible le déchiffrement. Dans ce cas, les opérations de codages et de décodages ne sont pas de même nature, d'où le qualificatif d'« asymétrique ».

Facile, difficile

Un cryptosystème à clef publique s'appuie sur une fonction dite à sens unique. Essentiellement, il s'agit d'une fonction f telle que :

il est facile d'évaluer $f(x)$ lorsqu'on connaît x dans l'espace de définition de f ;

connaissant un y dans l'espace d'arrivée, il est difficile de trouver un x tel que $f(x) = y$.

Comme mentionné précédemment, pour le cryptosystème RSA, la fonction à sens unique consiste simplement en la multiplication des entiers.

Logarithme discret

Le logarithme discret est un objet mathématique utilisé en cryptologie. C'est l'analogue du logarithme réel qui est la réciproque de l'exponentielle, mais dans un groupe cyclique G fini.

Le logarithme discret est utilisé pour la cryptographie à clé publique, typiquement dans l'échange de clés Diffie-Hellman et le chiffrement El Gamal. La raison est que, pour un certain nombre de groupes, on ne connait pas d'algorithme efficace pour le calcul du logarithme discret, alors que celui de la réciproque, l'exponentiation, se réalise en un nombre de multiplications logarithmique en la taille de l'argument (voir exponentiation rapide)

Définition

On considère un groupe cyclique G fini d'ordre n (dont la loi est notée multiplicativement) et un générateur b de G. Alors chaque élément x de G peut être écrit sous la forme $x = b_k$ pour certains entiers k ; de plus, deux quelconques de ces entiers sont nécessairement congrus modulo n. Le logarithme discret peut être défini comme le plus petit entier naturel k vérifiant cette propriété (il en existe un seul tel que $0 \leq k < n$). C'est donc une application réciproque de l'exponentiation $k \mapsto b_k$.

Resumé - Texte A

Section 1

Le texte propose d'étudier un protocole de mise en commun de secrets utilisant des matrices à corps fini. Le fonctionnement de cette approche se base sur la difficulté de calculer mes images réciproques de quelques éléments dans un temps raisonnable. L'énoncé du problème est le suivant : Étant donné que l'on se mette d'accord sur un groupe G et un élément μ de G, et que A et B échangent leurs contribution au secret $s_A = \mu^a$ et $s_B = \mu^b$,leur secret commun et dont disposent A et B uniquement est $(\mu^a)^b = (\mu^b)^a = \mu^{ab}$. Le but du texte est de trouver une solution du problème dit : "Logarithme discret" et proposer des recommandations pour les choix de a et b afin d'éviter un espionnage.

Section 2

Cette section,dont les lemmes et le theorème vont être démontrés dans notre mémoire,à pour finalité d'éviter un brute forcing du secret commun.Par consequent,A et B se doivent de :

- choisir a et b tq $a, b \ > deg(\Pi_M)$
- choisir Π_A, Π_B premiers à Π_M

Section 3

cette section s'intéresse à l'avantage de résoudre le problème du logarithme discret via un logarithme de reduction dans le calcul matriciel. Le but étant de trouver a telque $M^a = A$. Nous adoptons la démarche suivante en ayant pour but de determiner $a\ mod(ordre(M))$ en utilisant la proposition 1.

1- On essaie de trouverr un $\lambda \in Sp(M)$ associé au vecteur propre V. On peut donc trouver une base N ayant pour premier vecteur V, telque :

$$M = NPN^{-1}$$

avec $P = \begin{pmatrix} \lambda & \\ \vdots & J \\ 0 & \end{pmatrix}$

Ainsi $A = M^a = N \begin{pmatrix} \lambda^a & \\ \vdots & J^a \\ 0 & \end{pmatrix} N^{-1}$

On retrouve directement $a\ mod(ordre(\lambda)) \quad (1)$

2- On prend Π_M le pôlynome minimal de M et on choisit son facteur de multiplicité maximale t. Soit λ sa valeur propre associée à v, ainsi :

$$N^{-1}MN = \begin{pmatrix} J & . \\ 0 & . \end{pmatrix}$$

$$N^{-1}AN = \begin{pmatrix} J^a & . \\ 0 & . \end{pmatrix}$$

Nous retrouvrons alors $a[p] = u \quad (2)$

On sait de plus que :

$$a[ord(M)] = a[p] + pv$$

$$AM^{-u} = M^a M^{-u} = M^{a-u} = M^{a[ord(M)-a[ord(M)+pv} = (M^P)^v$$

On retrouve en répétant ce procédé au max $[\frac{ln(t)}{p}]$ fois.

La valeur $a[p^\tau] \quad (3)$

La proposition (1) nous livre par ricochet $a[ord(M)]$

Section 4

Cette section est une extension de la section 3 qui se focalise sur un sous-ensemble de matrices particulieres qui est celui des inversibles définis par :

$$M(a_0, \cdots, a_{n-1}) = \begin{pmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_{n-1} & a_0 & \cdots & a_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & a_0 \end{pmatrix}$$

Preuves

Preuve $Lemme\ 1$**:**

Supposons que $a < deg(\Pi_M)$

La division euclidienne de X^a sur Π_M nous livre : $\exists (Q, f) \in \mathbb{R}[X] : X^a = Q \cdot \Pi_M + f$ avec $deg(f) < deg(\Pi_M)$

Distinguons deux cas:

Si $Q = 0$, alors : $X^a = f$. Ce qu'il faut démontrer

Si $Q \neq 0$, alors :

$deg(X^a) \leqslant max(deg(Q) + deg(\Pi_M), deg(f))$

or $deg(f) < deg(\Pi_M)$

donc $deg(f) < deg(\Pi_M) + deg(Q) \implies a = deg(Q) + deg(\Pi_M) \implies a - deg(\Pi_M) = deg(Q)$

or d'après l'hypothèse, on a : $a < deg(\Pi_M)$

donc $deg(Q) < 0$ donc $Q = 0$

ce qui est contradictoire avec notre hypothèse

ainsi $Q = 0$ et finalement $a < deg(\Pi_M) \implies X^a = f$

Preuve $Lemme\ 2$**:**

Supposons que $\Pi_B | \Pi_M$

On a $\exists Q \in \mathbb{R}[X] : X^a = Q \cdot Q' \cdot \Pi_B + f$

En B, on a : $B^a = (Q \cdot Q' \cdot \Pi_B)(B) + f(B) = f(B)$. Car $\Pi_B(B) = 0$

donc $(M^b)^a = B^a = f(B)$

d'où : $f(B) = M^{ab}$

Preuve $Theorème\ 1$**:**

Soit $P_B = ppcm(\Pi_M, \Pi_B)$

La division Euclidienne de X^a par P_B nous livre : $\exists!(R,Q) \in \mathbb{R}^2[X]$ tq : $X^a = Q \cdot P_B + R$ avec $deg(R) < deg(P_B)$

En M on a : $A = (Q \cdot P_B)(M) + R(M) = R(M)$

En $M^b = B$ on a : $M^{ab} = (Q \cdot P_B)(B) + R(B) = R(B)$

Car $P_B = ppcm(\Pi_M, \Pi_B) \implies P_B(M) = P_B(B) = 0$

D'où l'existance et l'unicité de $g = R$

En effet on a :

$X^a = Q \cdot \Pi_M + f$ avec $deg(f) < deg(\Pi_M)$

$X^a = Q' \cdot P_B + g$ avec $deg(g) < deg(P_B)$

et $P_B = ppcm(\Pi_M, \Pi_B) \implies \Pi_M | P_B$

donc $\exists Q''$ tq $X^a = Q' \cdot Q'' \cdot \Pi_M + g = Q \cdot \Pi_M + f$

Alors : $g = f + (Q - Q' \cdot Q'')\Pi_M$

d'où $\exists h = Q - Q' \cdot Q''$ tq : $g = f + \Pi_M h$

Avec $deg(\Pi_M h) = deg(g - f) \leq max(deg(g), deg(f))$

Or $deg(g) < deg(P_B)$ et $deg(f) < deg(\Pi_M) \leq deg(P_B)$, donc $max(deg(g), deg(f)) < deg(P_B)$

C'est-à-dire $deg(h) + deg(\Pi_M) < deg(P_B) \implies deg(h) < deg(P_B) - deg(\Pi_M)$.

Applications numériques

À l'aide des méthodes de la Section 2 peut-on calculer le secret commun lorsque $p = 29$?

In [3]:
```
p=29; n=3
Zp=Zmod(p)
```

In [4]:
```
Mp=MatrixSpace(Zp,n,n);Mp
```

Out[4]:
```
Full MatrixSpace of 3 by 3 dense matrices over Ring of integers modulo 29
```

In [5]:
```
from sage.all import *
R = ZZ['x']
x = R.gen()
f = x**2 - 1
g = x**5
print g.degree(x)
h = f.mod(x**2 - 1)
print h.degree(x)
print g.quo_rem(f)
```

```
5
-1
(x^3 + x, x)
```

In [6]:
```
M = Mp.matrix([7,4,12,13,19,10,15,9,26])
Pm=M.minimal_polynomial()
M,Pm
```

Out[6]:
```
(
[ 7  4 12]
[13 19 10]
[15  9 26], x^3 + 6*x^2 + 23*x + 27
)
```

In [7]:
```
A = Mp.matrix([14,7,4,9,3,17,12,4,6])
Pa=A.minimal_polynomial()
A,Pa
```

Out[7]:
```
(
[14  7  4]
[ 9  3 17]
[12  4  6], x^3 + 6*x^2 + 23*x + 27
)
```

In [8]:
```
B = Mp.matrix([8,6,4,15,15,18,21,26,23])
Pb=B.minimal_polynomial()
B,Pb
```

Out[8]:
```
(
[ 8  6  4]
[15 15 18]
[21 26 23], x^3 + 12*x^2 + 7*x + 22
)
```

Test lemme 1 :

In [11]:
```
#Pour A :
for i in range (0,30):
    if M^i == A:
        a =i
        print('a = ',a)
if a < Pm.degree():
    print " f = X^a"
else:
    print " Lemme 1 non verifié"
```

```
('a = ', 29)
 Lemme 1 non verifié
```

In [12]:
```
#Pour B :
for i in range (0,30):
    if M^i == B:
        b =i
        print('b = ', b)
if b < Pm.degree():
    print " f = X^b"
else:
    print " Lemme 1 non verifié"
```

```
('b = ', 12)
 Lemme 1 non verifié
```

In [13]:
```
a_fois_b = a*b; a_fois_b
```

Out[13]:
```
348
```

Test lemme 2 :

In [14]:
```
#Pour A :
r = Pm%Pa
if r==0:
    print " f(A) = M^ab"
else:
    print " Lemme 2 non verifié"
```

```
 f(A) = M^ab
```

In [15]:
```
#Pour B :
r = Pm%Pb
if r==0:
    print " f(B) = M^ab"
else:
    print " Lemme 2 non verifié"
```

```
 Lemme 2 non verifié
```

In [16]:
```
ppb = lcm(Pm,Pb); ppb
```

Out[16]:
```
x^6 + 18*x^5 + 15*x^4 + 19*x^3 + 8*x^2 + 28*x + 14
```

Test théorème :

Nous supposons maintenant que nous ne disposons pas de a et b et nous essaierons de trouver le polynome g telque $A = g(M)$ et $M^{ab} = g(B)$ qui est le message commun tant attendu (pour un espion)

In [18]:
```
# Puisque la condition du lemme 1 n'est pas vérifiée pour a et b alors a > deg(Pm)
Liste_g = []
Pm_degree = 3
c=0
for i in range(Pm_degree,30):
    R = ZZ['x']
    x = R.gen()
    g = x**i
    Liste_g.append(g.quo_rem(ppb)[1])
    z= g.quo_rem(ppb)[1]
    if z(M)== A :
        print(z)
        print(i)
        c=c+1

if c==1 :
    print('Lunicite du polynome trouve z est verifiee ', c)
```

```
15*x^5 + 19*x^4 + 9*x^3 + 12*x^2 + 8*x + 15
29
('Lunicite du polynome trouve z est verifiee ', 1)
```

In [19]:
```
print(z(B))
if z(B) == M^a_fois_b:
    print('Décryptage réussi !')
```

```
[19  5 28]
[ 8  9 18]
[16 10 18]
Décryptage réussi !
```

De même, peut-on trouver un ensemble dematrices le plus petit possible qui contienne le secret commun lorsque $p = 29$

In [20]:
```
M= Mp.matrix([5,1,2,26,23,3,16,21,20])
A = Mp.matrix([6,24,19,15,3,14,7,11,18])
B= Mp.matrix([7,19,9,1,1,28,14,22,2])
Pm=M.minimal_polynomial()
Pa=A.minimal_polynomial()
Pb=B.minimal_polynomial()

print(Pm,Pa,Pb)
```

```
(x^2 + 17*x + 6, x^2 + 27*x + 5, x^2 + 6*x + 25)
```

Test lemme 1 :

In [22]:
```
#Pour A :
for i in range (0,30):
    if M^i == A:
        a =i
        print('a = ',a)
if a < Pm.degree():
    print " f = X^a"
else:
    print " Lemme 1 non verifié"
```

```
('a = ', 13)
('a = ', 27)
 Lemme 1 non verifié
```

In [23]:
```
#Pour B :
for i in range (0,30):
    if M^i == B:
        b =i
        print('b = ', b)
if b < Pm.degree():
    print " f = X^b"
else:
    print " Lemme 1 non verifié"
```

```
('b = ', 12)
('b = ', 26)
 Lemme 1 non verifié
```

Remarquons que les (a,b) ne sont pas uniques, nous disposons de 4 multiplications possibles pour a*b. Un calcul ultérieur nous montrera que malgrès ces 4 calculs possibles produits le calcul de M^ab donne la meme matrice..

Test lemme 2 :

In [24]:
```
#Pour A :
r = Pm%Pa
if r==0:
    print " f(A) = M^ab"
else:
    print " Lemme 2 non verifié"
```

```
 Lemme 2 non verifié
```

In [25]:
```
#Pour B :
r = Pm%Pb
if r==0:
    print " f(B) = M^ab"
else:
    print " Lemme 2 non verifié"
```

```
 Lemme 2 non verifié
```

In [26]:
```
ppb = lcm(Pm,Pb); ppb
```

Out[26]:
```
x^3 + x^2 + 24*x + 20
```

Test théorème :

Nous supposons maitenant que nous ne disposons pas de a et b et nous essaierons de trouver le polynome g tq $A = g(M)$ et $M^{ab} = g(B)$ qui est le message commun tant attendu (pour un espion)

In [28]:
```
# Puisque la condition du lemme 1 n'est pas vérifiée pour a et b alors a > deg(Pm)
Liste_g = []
Pm_degree = 3
c=0
for i in range(Pm_degree,30):
    R = ZZ['x']
    x = R.gen()
    g = x**i
    Liste_g.append(g.quo_rem(ppb)[1])
    z= g.quo_rem(ppb)[1]
    if z(M)== A :
        print(z)
        print(i)
        c=c+1

if c==2 :
    print('Lunicite du polynome trouve z est verifiee (meme polynome) ', c)
```

```
13*x^2 + 13*x + 22
13
13*x^2 + 13*x + 22
27
('Lunicite du polynome trouve z est verifiee (meme polynome) ', 2)
```

In [44]:
```
p=29
Zp=Zmod(p)
print(z(B))
c=0
for i in [12*13,12*27,26*13,26*27]:
    print("\n")
    print(i, Zp(i))
    print(M^i)
    if z(M^i)==z(B):
        c=1

if c==Zp(1):
    print('Décryptage réussi !')
else :
    print('Décryptage tenu à lechec !')
```

```
[ 7 19  9]
[ 1  1 28]
[14 22  2]


(156, 11)
[16 23  0]
[ 4 22 22]
[12 16 11]


(324, 5)
[13 27  4]
[ 3 14  1]
[21 24 22]


(338, 19)
[11 25 20]
[11 10 12]
[ 6 17  3]


(702, 6)
[ 0  1 25]
[17  0 22]
[10 27 24]
Décryptage tenu à lechec !
```

Nous avons trouvé un cas de figure où l'ensemble des matrices trouvées ne comprennent pas le message commun. Ceci était attendu vu que la fonction retrouvée peut vérifier g(M)=A et ne pas vérifier g(B)=M^ab (les fonctions polyomiales peuvent coincider dans un nombre de points inferieurs à leurs degres sans pour autant etre égales).

Quel est l'ordre de la matrice de $GLn(F29)$ suivante

Afin de déterminer l'ordre de la matrice donnée, nous allons utiliser l'équation (3) de la propostion 1. Ainsi, nous allons :

1 - déterminer le polynome minimal Pm, le factoriser et identifier ses racines dans Zp

2 - déterminer le ppcm des ordres de ces racines

3 - retrouver le réel tau vérifiant l'inégalité, nous allons procéder à l'implementation de l'algorithme de la dichotomie cependant, cette démarche ne nous livre pas l'unicité du réel tau.

En conséquence, un petite reflexion suivant le raisonnement par recherche et synthèse nous invite à trouver un réel tau tq : p^tau est un naturel (le plus proche possible) et lancer une boucle qui nous permet de tester ord(M) suivant l'équation (3).

(Résultat final : $ord(M) = 140$)

In [1]:
```
import numpy as np
```

In [34]:
```
M = Mp.matrix([8,26,0,2,11,11,6,8,20])
Pm=M.minimal_polynomial();Pm
```

Out[34]:
```
x^3 + 19*x^2 + 9*x + 8
```

In [37]:
```
Pm.factor()
```

Out[37]:
```
(x + 2) * (x^2 + 17*x + 4)
```

In [38]:
```
Pm.roots()
```

Out[38]:
```
[(27, 1)]
```

In [43]:
```
root=Zp(27)
ordre=root.multiplicative_order()
```

In [99]:
```
def f(x):
    return ln(29)*x
def g(x):
    return f(x-1)

def Dichotomie(h,k,a,b,e):
    debut = a
    fin = b
    #calcul de la longueur de [a,b]
    ecart = b-a
    while ecart > e:
        #calcul du milieu
        m = (debut+fin)/2
        if h(m) >= k and h(m-1) < k:
            #la solution est inférieure à m
            fin = m
        else:
            #la solution est supérieure à m
            debut = m
        print(m)
        ecart = fin-debut
    return m
```

In [146]:
```
potential_tau = []
for tau in np.linspace(0,1,1000):
    if 29**(tau) - int(29**(tau)) <= 10**(-2):
        potential_tau.append(tau)
potential_tau
```

Out[146]:
```
[0.0,
 0.001001001001001001,
 0.002002002002002002,
 0.2062062062062062,
 0.2072072072072072,
 0.3263263263263263,
 0.4124124124124124,
 0.47847847847847846,
 0.5325325325325325,
 0.6176176176176176,
 0.6526526526526526,
 0.7617617617617618,
 0.7837837837837838,
 0.955955955955956,
 1.0]
```

In [148]:
```
identity = Mp.matrix([1,0,0,0,1,0,0,0,1])

for e in potential_tau[1:]:
    ordre_m=int(p^e)*ordre
    if M^ordre_m == identity:
        print(ordre_m)
```

```
140
700
```

In [56]:
```
M.multiplicative_order()
```

Out[56]:
```
140
```

Le cout du calcul des valeurs propres de La matrice M

La plupart des algorithmes de calcul des valeurs propres d'une matrice proposent une complexité de O(n^3). Or, pour notre cas la matrice M est assez particulière. On trouve que les valeurs propres de M sont données directement par la relation ci-dessous :

$$\lambda_j = c_0 + c_{n-1}\omega^j + c_{n-2}\omega^{2j} + \ldots + c_1\omega^{(n-1)j}, \quad j = 0, 1, \ldots, n-1.$$

D'ou la complexité du calcul des valeurs propres de M est *quadratique*

In [156]:
```
T=Mp.matrix([1,8,5,5,1,8,8,5,1]);T
```

Out[156]:
```
[1 8 5]
[5 1 8]
[8 5 1]
```

In [159]:
```
T.determinant()
```

Out[159]:
```
25
```

```
In [162]: Pt = T.characteristic_polynomial()

In [163]: Pt.factor()
Out[163]: (x + 15) * (x^2 + 11*x + 8)

In [164]: Pt.roots()
Out[164]: [(14, 1)]
```

Remarque :

Il est difficile de calculer les valeurs propres de M, car les built-in méthodes ne sont pas concues à travailler avec les matrices à coefficients dans Fp.

Complexité des operations

Section 2 :

Notre hypothèse stipule que : $deg(\Pi_M < \ a, b < \ p$ ainsi il faut tester tout les $\ i \in [deg(\Pi_M, p]$

$g_i := Reste(X^i, P_B)$
$g_2 := Reste(X^i, P_A)$

Ainsi la complexité du calcul du theorème 1 est :

$\Theta((p - deg(\Pi_M))^2 . min(deg(P_A, deg(P_B)))$

Section 3 :

Nous detaillerons dans ce qui suit les étapes intermédiaires pour le calcul de $'ord(M)'$ et leur complexité

• Calculer pôlynome minimal en utilisant l'algorithme de Wiedemam $\implies \Theta(n^3)\ ;\ n = taille(M)$

• Décomposer en facteurs irréductibles en utilisant l'algorithme de Berlekamp $\implies \Theta(n^3)\ ;\ n = taille(\Pi_M)$

• Calcul de l'ordre des racines

• Calcul du ppcm des ordres à l'aide de l'algorithme d'Euclide $\implies \Theta(n^3)\ ;\ n = cardinal(racines)$

• Retrouver τ, vérifiant $\ p^{\tau-1} \leq max \leq p^{\tau}$

Ainsi la complexité générale de cette proposition est cubique en la taille de la matrice M .

Fin